

James C. SOLINSKY
Serial No. 09/658,275

Remarks

Reconsideration and allowance of the subject patent application are respectfully requested.

Claims 1-3, 5-8, 9-11, 13-19, 21-24 and 29-32 were provisionally rejected under the judicially-created doctrine of obviousness-type double patenting based on certain claims of co-pending Application No. 09/658,276. In a paper filed on even date herewith, the '276 application is being expressly abandoned for the purpose of avoiding double patenting issues with respect to the claims of the '276 and of the subject patent application. Accordingly, the provisional double patenting rejection of claims 1-3, 5-8, 9-11, 13-19, 21-24 and 29-32 is moot.

Claims 5-8, 13-16 and 48-51 were objected to because they are alleged to be apparatus claims that depend from method claims. Applicant believes these claims include every limitation of the claims from which they depend as required by 35 U.S.C. Section 112, and, as expressly noted in MPEP Section 608.01(n), "[t]he fact that the independent and dependent claims are in different statutory classes does not, in itself, render the latter improper." Nonetheless, to expedite prosecution, claims 5-8, 13-16 and 48-51 have been rewritten as method claims.

Claims 6-8, 14-16, 22-24, 29-31 and 49-51 were rejected under 35 U.S.C. Section 112, first paragraph, as allegedly failing to reasonably provide enablement for an integrated circuit or hardware processing engine. Specifically, the office action states that although the specification is enabling for a mathematical algorithm, it is allegedly not sufficient to enable any person skilled in the art to provide the invention in an integrated circuit or hardware processing engine. Applicant respectfully traverses this rejection.

As previously discussed, with the advent of mathematical languages, such as Mathematica[®], mathematical equations can be directly converted to computer processing code algorithms. In addition, using Handel-C[®], compilable C code can be converted into hardware net-lists. Equations (1)-(4) on page 27, equation (10) on page 29, and equations (11)-(17) on pages 31-32, for example, all use standard algebra and vector representations that are common elements in descriptions either directly translatable into, for example, Mathematica[®] software, or commonly used in translation to MatLab[®] software environments. Applicant respectfully

James C. SOLINSKY
Serial No. 09/658,275

submits that a person skilled in the art would be readily able to implement the teachings of this application into hardware engines or integrated circuits or net lists.

As example evidence, Applicant attaches hereto a copy of a web page (updated February 2000 and identifying Christian Peter of Oxford University as the author) entitled: "Overview: Hardware Compilation and the Handel-C language." This web page notes:

Handel-C is a programming language designed for compiling programs into hardware images of FPGAs or ASICs. It is basically a small subset of C, extended with a few constructs for configuring the hardware device and to support generation of efficient hardware. It comprises all common expressions necessary to describe complex algorithms, but lacks processor-oriented features like pointers and floating point arithmetic. The programs are mapped into hardware at the netlist level, currently in xnf or edif format.

In contrasting Handel-C with the VHDL hardware description language, the web page further notes:

The low-level problems are hidden completely, all the gate-level decisions and optimisation are done by the compiler so that the programmer can focus his mind on the task he wants to implement. As a consequence, hardware design using Handel-C resembles more to programming than to hardware engineering, and in fact, this language is developed for programmers who have no hardware knowledge at all.

Applicant respectfully submits that one of ordinary skill in the art reading the disclosure of the subject application would have been readily able to develop hardware engines or integrated circuits or netlists based on the specification without undue experimentation. As discussed above, such designs could be readily implemented using well-known tools commercially available at the time the application was filed, at least one these tools (Handel-C) being developed for use by programmers "who have no hardware knowledge at all."

Applicants believe the subject matter of claims 6-8, 14-16, 22-24, 29-31 and 49-51 is fully enabled by the disclosure as originally filed and respectfully request withdrawal of the rejection of claims 6-8, 14-16, 22-24, 29-31 and 49-51 based on Section 112, first paragraph.

Claims 1-52 were rejected under 35 U.S.C. Section 112, second paragraph, as allegedly being indefinite. Applicant has amended the independent claims in light of the Examiner's comments to more particularly describe that the outputs are output signals. Based on these

James C. SOLINSKY
Serial No. 09/658,275

amendments, withdrawal of the rejection of claims 1-52 under 35 U.S.C. Section 112, second paragraph, is respectfully requested.

Applicant acknowledges with appreciation the indication that claims 35-52 recite allowable subject matter and that the amendments to the other claims to incorporate features argued, but not claimed, would be favorably considered if the Section 112 and 101 issues were resolved.

In this regard, claims 1, 9, 17, 25 and 32 have been amended to specify "a user memory model" in the context of the claimed systems and methods. This feature is believed to distinguish over the applied documents and any combinations thereof.

Claims 35-52 are believed to be allowable based on the indication to this effect in the office action and the addressing of the Section 112 and Section 101 issues.

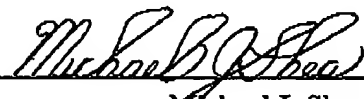
New claims 53-56 have been added. The subject matter of these new claims is fully supported by the original disclosure and no new matter is added. Claims 53 and 54 depend from claim 35 and claims 55 and 56 depend from claim 52. These claims are believed to be allowable at least because of their respective dependencies from either claim 35 or 52.

James C. SOLINSKY
Serial No. 09/658,275

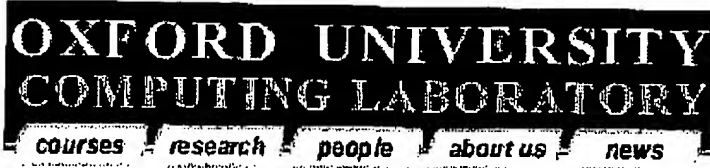
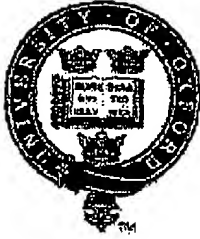
The pending claims are believed to be allowable and favorable office action is respectfully requested. Should any issues remain, the Examiner is invited to telephone the undersigned at the number listed below.

Respectfully submitted,

NIXON & VANDERHYE, P.C.

By: 
Michael J. Shea
Reg. No. 34,725

MJS:mjs
901 North Glebe Road, 11th Floor
Arlington, VA 22203-1808
Telephone: (703) 816-4000
Facsimile: (703) 816-4100



Overview: Hardware Compilation and the Handel-C language

Christian Peter, Oxford University Computing Laboratory, UK
cpeter@comlab.ox.ac.uk

COPY

Hardware compilation [Page 1996] is a powerful tool for hardware/software codesign. Conventionally, it is common to draft a system using a high-level programming language, like C, and then realise the hardware part of that system using a hardware description language like VHDL [Perry 1994]. Hardware compilation using Handel-C [Page 1996] simplifies the design process, since it provides the convenience of a C-like high-level programming language to generate the hardware as well.

Handel-C is a programming language designed for compiling programs into hardware images of FPGAs or ASICs. It is basically a small subset of C, extended with a few constructs for configuring the hardware device and to support generation of efficient hardware. It comprises all common expressions necessary to describe complex algorithms, but lacks processor-oriented features like pointers and floating point arithmetic. The programs are mapped into hardware at the netlist level, currently in xnf or edif format.

Opposed to other approaches of high-level language hardware design, which actually use C to describe the behaviour and simply translate it into a netlist, Handel-C targets hardware directly, and provides a few hardware optimising features. A big advantage, compared to the C-translators, is that variables and constants can be given a certain width, as small as one bit. When using C as the describing language, the smallest integer is possibly 8 bits wide, if not 16, which means that one wastes at least 7 registers when declaring a simple flag. Also, Handel-C provides bit manipulation operators and the possibility of parallel processing of single statements or whole modules. This can not be realised with other approaches basing on a sequential language.

COPY

Comparing Handel-C with VHDL shows that the aims of these languages are quite different. VHDL is designed for hardware engineers who want to create sophisticated circuits. It provides all constructs necessary to craft complex, tailor made hardware designs. By choosing the right elements and language constructs in the right order, the specialist can specify every single gate or flip-flop built and manipulate the propagation delays of signals throughout the system. On the other hand, VHDL expects that the developer knows about low-level hardware and requires him continuously thinking about the gate-level effects of every single code sequence. This quite easily distracts from the actual algorithmic or functional subject and ties up a lot of the designer's attention.

In contrast to that, Handel-C is not designed to be a hardware description language, but a high-level programming language with hardware output. It doesn't provide highly specialised hardware features and allows only the design of digital, synchronous circuits. Instead of trying to cover all potentially possible design particularities, its focus is on fast prototyping and optimising at the algorithmic level. The low-level problems are hidden completely, all the gate-level decisions and optimisation are done by the compiler so that the programmer can focus his mind on the task he wants to implement. As a consequence, hardware design using Handel-C resembles more to programming than to hardware engineering, and in fact, this language is developed for programmers who have no hardware knowledge at all.

That Handel-C is designed for fast prototyping can also be seen when working with the simulator. This small but efficient tool simulates the behaviour of the circuit at the algorithmic level, based on the semantics of the Handel-C program. Compared with hardware simulators, which usually analyse the gate-level function of the circuit, the simulation time is very short (seconds opposed to several minutes). Together with the very fast compilation, this encourages the designer to try out several implementations of the design.

To sum up, hardware design with Handel-C is very much like programming. Unlike with hardware description languages, the designer is not confronted with gate-level problems like fan-in and fan-out or choosing the appropriate type of gates or registers to be used. Apart from freeing the programmer's mind from those low-level decisions, it is much faster and more convenient to describe the system's desired behaviour at the algorithmic level. The fast compilation, combined with the high-level simulator, allows to try out several implementation strategies within a very short time.

[Page 1996] Page, I. : Constructing hardware-software systems from a single description ;

Journal of VLSI Signal Processing, 12(1), pp. 87-107, 1996.

[Perry 1994] Perry, Douglas L. : VHDL, 2nd edition ;

McGraw-Hill series on computer engineering) ; New York, 1994

[back to text](#)

COPY

[Home](#)



[oucl](#) [work](#) [christian.peter](#)

Updated February 2000

[Home](#) | [Search](#) | [SiteMap](#) | [Feedback](#)

© OUCL, 1994-2000

[Courses](#) | [Research](#) | [People](#) | [About us](#) | [News](#)